WP526 (v1.0)

# Importance of Programmability in Next-Generation Security Appliances

*Traditional methods of deploying network security through software-based firewalls do not scale because the latency and bandwidth requirements cannot be addressed. The flexibility and configurability of Xilinx adaptive devices combined with IP and tool offerings significantly improves security processing performance.*

**ABSTRACT**

This white paper explores multiple firewall architectures, which include software- and NPU-based architectures and explains why next-generation designs need an inline firewall architecture using Xilinx's adaptive devices.

Xilinx's 16nm FPGAs and SoCs and 7nm Versal™ ACAPs offer multiple architectural components in the form of hardened blocks and soft IP, which make them ideal for designing next-generation security appliances. These IPs include high-speed SerDes and multirate interface IP, such as hardened MAC, PCIe® interfaces, and memory controllers. Xilinx devices also offer the latest state-of-the-art memory architecture with soft search IPs for flow classification, making them best suited for network security and firewall applications.

# Introduction

This white paper describes the functionality, deployment, and architecture of security appliances acting as next-generation firewalls (NGFWs) in enterprise and telco data center networks. The flexibility and configurability of Xilinx devices combined with IP and tool offerings can significantly improve the performance of network security appliances used for threat detection and prevention in networks and scale for performance. These devices can also implement upcoming next-generation security technologies such as post quantum crypto (PQC) and machine learning (ML) for anomaly detection.

Since enterprise networks are transitioning towards policy and intent-based networking, the flows and policies define the actions (route, QoS, discarding, tagging etc.) on traffic. Also, the required security policies for incoming traffic continuously change based on the nature of flows in a network. Most traffic requires the processing of network traffic flows in a dynamic and stateful manner.

Traditional port-based firewalls provide perimeter-based protection, which filters the traffic based on packet parameters (e.g., IP addresses and TCP/UDP port numbers) because application awareness is only handled in the software, which cannot scale in performance. The NGFWs should not just be able to identify and act on a specific class of traffic but should also be able to identify the threat associated in the content within an application. Many of the applications used by enterprises allow port hopping, use nonstandard ports, or hide the threats in SSL tunnels so threats and malware cannot be detected by traditional static port-based firewalls.

# Firewalls in Enterprise Networks

In previous generations, network firewalls were deployed at the network edges to ensure security between enterprise offices, connected using multiple transport network technologies, and often using the same network pipe as a public network. With the evolution of policy-based networks and the emergence of software-defined networks (SDN) and intent-based networks (IBN), the firewalls of different throughputs and features are being deployed at many different locations within the enterprise network. See Figure 1.

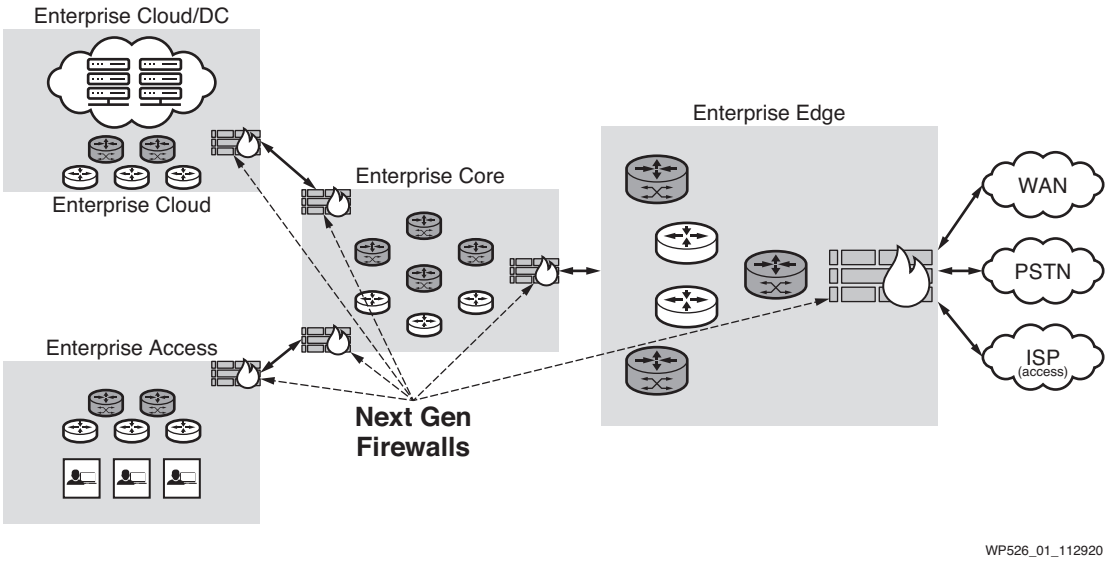


*Figure 1:* **Next-Generation Firewall in an Enterprise Network**

As shown in Figure 1, the role of firewalls has expanded from the perimeter of enterprise networks to multiple locations within an enterprise, e.g., connecting the main and branch offices of an enterprise and securing the connectivity edge, or securing the traffic of an enterprise data center from enterprise access. NGFWs can detect and prevent the threat and malware between the network segments based on multiple packet parameters (port, IP address, payload contents) or encryption technologies such as L3-VPN or SSL/TLS.

# Firewall Deployment and Functionality

Security appliances are responsible for inspecting and analyzing all the traffic entering from outside an enterprise network. Firewalls can be deployed at multiple locations within an enterprise network, e.g., traffic between different departments of an organization, or traffic entering from an enterprise access to an enterprise data center via multiple networks nodes consisting of switches and routers.

The security appliances (NGFW) can be deployed in inline or lookaside mode. The main difference between the two is that the inline mode connects directly to external network ports while the lookaside appliance can be connected to a tap or mirrored port of a switch or router. Figure 2 shows the firewall connectivity in a network. While the functionality of firewalls can be similar, the inline firewalls are more complex and have higher performance than the lookaside firewall appliances.

The scale and functionality of the firewalls deployed at specific places can differ in policy rules assignment, but some basic functionality (such as traffic classification, buffering etc.) remains the same.

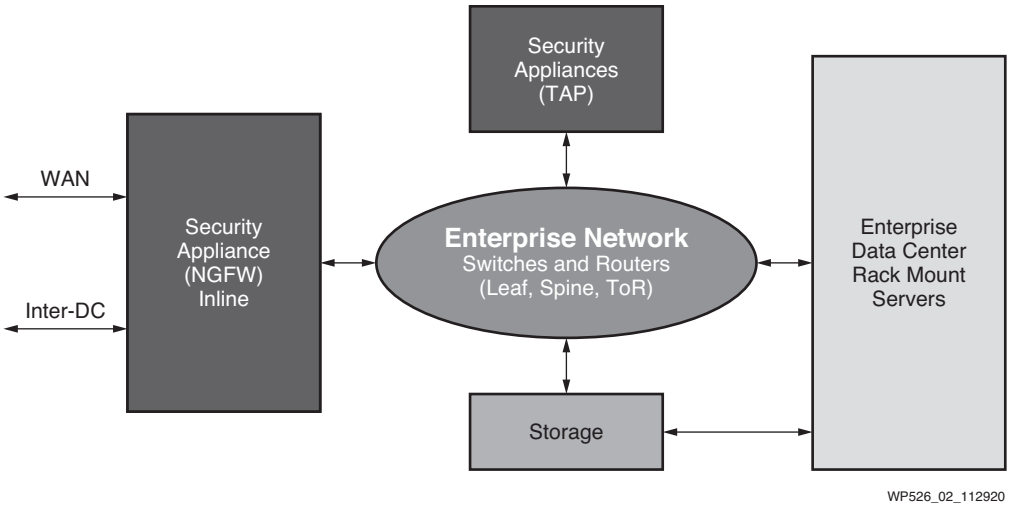


*Figure 2:* **NGFW in an Enterprise Network**

Network nodes or security appliances can be responsible for implementation of the following security functions:

1. L2 security - MACSec for link encryption
2. L3 security - VPN tunnels from users and other network nodes
3. Blocking and filtering of invalid traffic (protocol and port-based filtering)
4. TLS/SSL encryption/decryption of incoming and outgoing traffic
5. Anomaly detection across multiple traffic flows
6. Stateful pattern matching
7. Statistical anomaly detection
8. IP fragmentation
9. TCP reassembly and ordering
10. Signature/content matching using regular expression (regex)

Along with these features, the next-generation network security products have also started implementing ML models for network analytics and malware prediction. These models do not rely on traditional signature-based detection. The ML capable firewalls collect telemetry data and deploy the security policies ahead of the threat occurrence.

Some of the above functions are basic and part of any network node (secure switches and routers) and are implemented on deployed network switches and routers built using ASICs or programmable devices; others (L3 and beyond) are more complex and require significant traffic classification and processing. The complexity of traffic processing increases at higher network protocol layers. For example, Layer 1 (L1) security only needs encryption at the frame level (e.g., OTN transport payload frames), and is implemented in optical network nodes using bulk cipher protocol (AES-GCM). Layer 2 (L2) and layer 3 (L3) require packet processing at Ethernet and IP levels, which require packet-level processing. Layer 4 (L4) and beyond require content-level security processing where each TCP or UDP session consists of multiple Ethernet and IP packets. Some of

the L2 and L3 security functions can be easily implemented on hardware devices (ASICs, ASSPs, FPGAs, SoCs, ACAPs, and NPUs). Higher layer security processing (L3 and above) also needs the software-based content processing of incoming traffic for threat detection and mitigation.

Since the new access network technologies (5G fronthaul, PON, and cable) have significantly increased the throughput and flows exponentially in the last few years, the firewall appliances based only on software traffic processing are not enough to keep up with the performance and latency requirements at desired throughputs.

# Hardware Architecture of Next-Generation Firewall

Since the firewalls need to process and inspect all incoming traffic, they need to perform the following operations:

- L2/L3 packet processing
- L2/L3 security features
  - LinkSec/MACSec
  - L3-VPN/IPSec
- L4–L7 packet processing and security

The options for firewall design are shown in Figure 3.

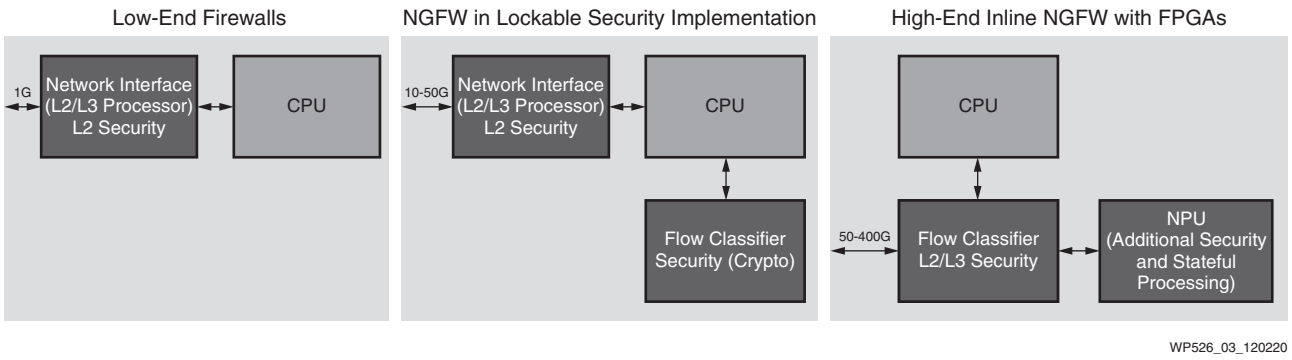


*Figure 3:* **Firewall Evolution: Lookaside vs. Inline Processing**

The low-end firewall appliances (usually below 10G) can be designed using a network interface device and CPU. Commonly existing network interface (NIC) devices (custom ASICs, FPGAs, or ASSPs) can process the incoming traffic (Ethernet and IP packets) and perform well known L2 and L3 functions while higher layer (L4–L7) functions are performed by the software running on the CPU.

Mid-range firewalls are capable of handling higher throughputs (10G–50G) and are mostly designed using a network interface device and a lookaside security processor (security ASICs, NPUs, or FPGAs). Since the software-only solution is not capable of classifying and processing traffic at higher throughputs, the lookaside security processor acts as the CPU co-processor to offload the functionality of encryption/description, public key infrastructure (PKI), and/or stateful flow processing. Though in this architecture the network interface can be an ASIC or NPU, use of an

FPGA for mid-grade firewalls is getting popular because it allows for the scalability and flexibility of processing incoming traffic in inline mode, which reduces the latency in threat detection and prevention.

High-end next-generation firewalls with 50G–400G throughputs are mostly designed for inline mode of operation. In inline mode, the network interface device needs to be more intelligent to process millions of flows, which involves deeper inspection of incoming and outgoing packets. This interface device also needs to implement security features, such as inline IPSec, with commonly used crypto protocols and TCP-level security. This architecture still uses an NPU for specific crypto protocols, PKI, and stateful processing. The flow classification requirements of the inline device vary in terms of number and complexity of flows and action taken at those flows at high throughputs. Consequently, the programmable devices (e.g., FPGAs) for inline security processing are ideal for implementing these functions. Compared to NPUs, FPGAs offer much more latency reduction and scalability in traffic processing. FPGAs are also now available with next-generation memory interfaces and on-chip high bandwidth memory (HBM), which is useful for memory intensive traffic processing applications.

# FPGAs as Flow Processors for Network Security

The traffic entering and leaving security appliances (firewalls) is encrypted at multiple levels. The L2 encryption/decryption (MACSec) is processed at link-level (L2) network nodes (switches and routers). The processing beyond L2 (MAC layer) most commonly includes deeper parsing, decryption of L3 tunnels (IPSec), and processing encrypted SSL traffic along with TCP/UDP flows. Packet processing involves parsing and classification of incoming packets and processing large number of flows (1–20M) at high throughputs (25–400Gb/s). It is not possible to implement traffic processing at these throughputs in software using CPU cores due to the number of compute resources (cores) required. NPUs can be used for relatively higher rate packet processing but cannot deliver low-latency, high-performance scalable flow processing because traffic is processed using the MIPS/RISC cores, and scheduling to those cores based on their availability is a challenge. The above limitations in CPU- and NPU-based architectures can be efficiently addressed using FPGA-based security appliances.

## Flow Processing for Security Appliances

Flow processing is a higher level abstraction of packet processing because a flow can consist of many packets of a similar type. Flow processing includes the following main components:

- Packet parsing
- Packet lookups
  - Route lookups
  - Flow lookups using wild card search on packet fields
- Packet editing
  - Checksum calculation
  - Packet header encapsulation/decapsulation
  - Security header encapsulation

Xilinx offers tools for packet processing using the higher layer abstraction language P4, which enables packet parsing, classification, lookup, and packet editing functions. Complete packet processing using P4 can be implemented at a higher layer of abstraction than needed for the RTL language-based implementation. The use of P4 adds additional flexibility to the already programmable FPGA architecture because it allows easy implementation of packet parsing, packet editing, and modifications of flow table entries.

As shown in Figure 4, P4 describes a packet processing pipeline architecture that can be compiled using the P4 compiler and mapped in Xilinx FPGA, using their basic architectural components. The P4 language defines packet parsing, lookups (IPv4, IPv6, and other packet fields), and editing (de-parsing) of the packets. The P4-defined architecture can be directly applied to the security processing pipeline such as IPSec security association (SAs), security policy (SP) lookups, and tunnel processing implementation of ingress/egress traffic.

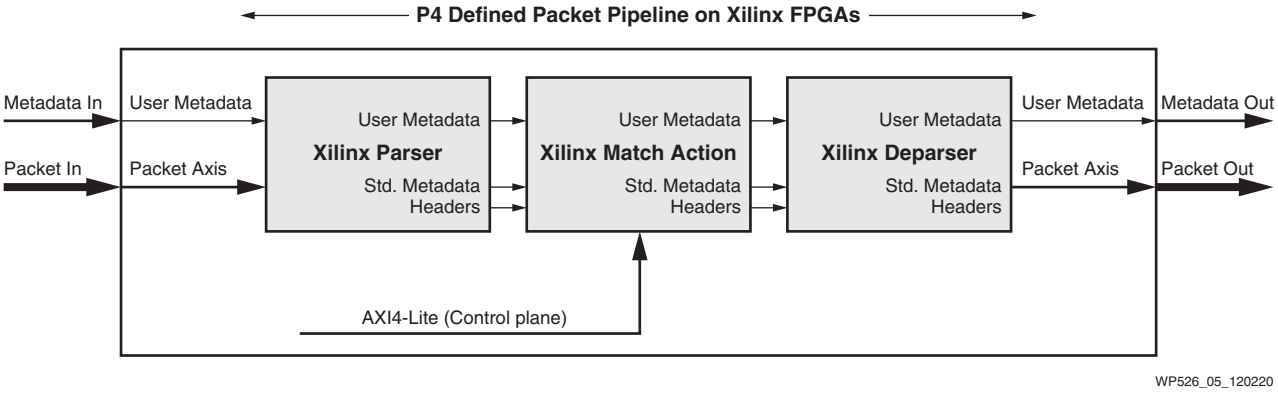


*Figure 4:* **P4-based Packet Processing Flow Classification and Lookup**

The three main components of packet processing include:

**Packet parsing:** Traffic originating from multiple applications, accessing the different nodes of enterprise and data center networks, need to be classified for different flows. Parsing involves extracting many packet parameters, which include the L2 headers, L3 headers, and fields from known offsets in the packet. These parsing requirements vary based on applications and signature located at different locations inside the packets. The flexible architecture of FPGAs along with the P4-defined parser can address these changing classification requirements.

**Packet lookups:**

After parsing, the packets need to be classified based on the type of traffic. The encrypted packets are decrypted based on protocol and security header fields.

The match-action module performs a lookup of the search key generated by the packet parser module for destination/action assignment.

For encrypted traffic, the key search includes determination of security association and security policies, which decides the decryption key information and policies to be applied on encrypted packets.

The L2 encrypted packet (MACSec) requires the simpler and direct lookup, while higher layer encryption lookups might be more complex, with wider keys and result values.

Figure 5 describes the example lookup for MACSec, IPSec, and TCP protocols. The number of these lookups can vary based on the network node, but in some cases, multi-layer lookups are needed for some traffic categories.

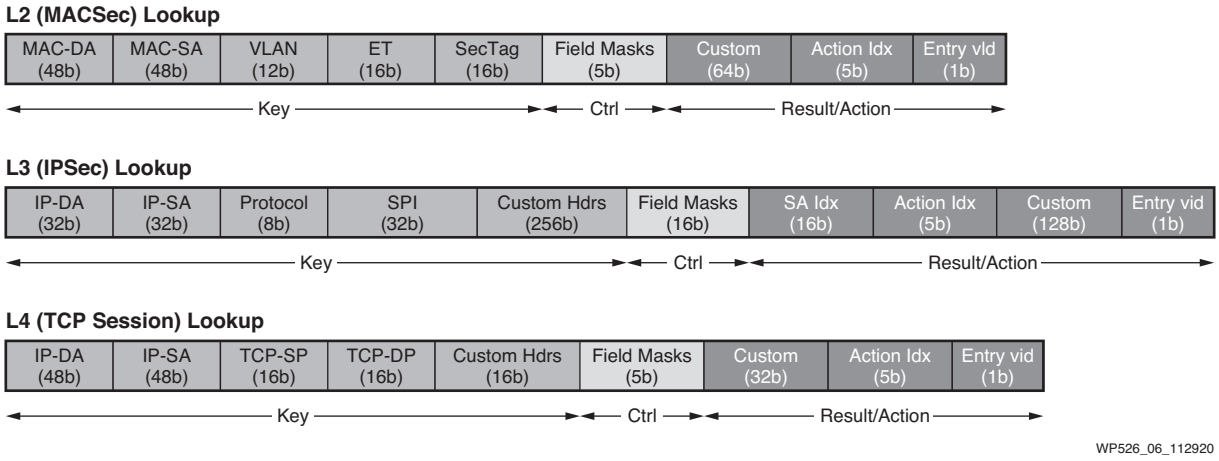


*Figure 5:* **Example Lookup for L2/L3/L4 Security Implementations**

The above lookups are specific to security processing. Firewalls can also implement additional lookups for router functionality, network address translation (NAT), and policy (or access control) lookups of incoming traffic (flows).

The following security and networking lookup categories include exact match, longest prefix match (LPM), and wild card search, using the keys made of packet header fields:

- Route lookups
- NAT lookups
- Flow classification with multiple fields

As shown in Figure 6, packet processing lookups are divided into three categories, with individual table and key size requirements.

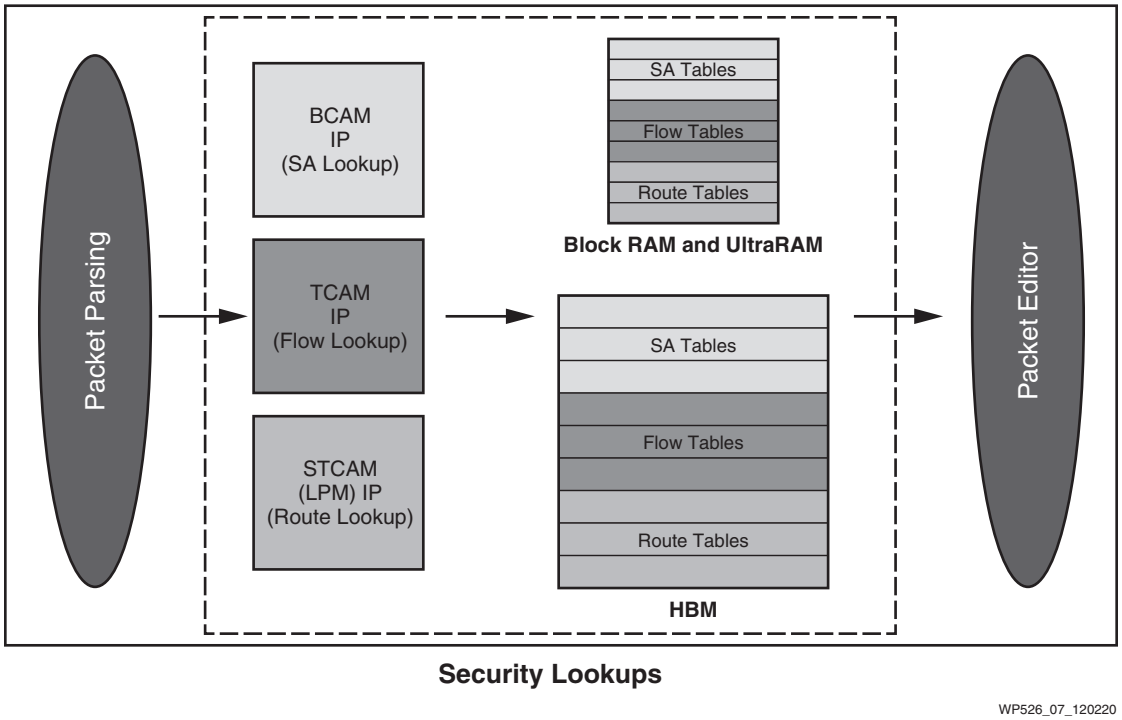


*Figure 6:* **Lookups Using FPGA-based Security Appliances**

Xilinx's IP portfolio includes search IPs for binary match, wild card ternary match and longest prefix match. These search IPs can be flexibly synthesized to fit into any Xilinx FPGA using on-chip SRAM and DRAM (HBM). All three categories of search IPs can work from 100Mb/s to 400Gb/s throughputs.

The key width, result width, and number of entries in a table decide the amount of on-chip logic resources and memory (SRAM/DRAM) used on the FPGA. Since Xilinx has a wide portfolio of device categories with different amounts of resources (logic/memory), users can select a Xilinx device with the appropriate resources for their throughput and table size requirements. The lookup IPs also come with application layer software APIs to modify and update the flow table entries.

**Packet Editing:**

Security processing involves sending the modified encrypted or decrypted packet out to the egress port after parsing, header lookup, and filtering. The outgoing traffic can include new and modified headers, updated packet fields, authentication headers, and error correction fields.

Some of the common packet processing requirements include:

- Changing L2/L3 headers (MAC/VLAN/IPv4/IPv6)
- Creating and updating the security (MACSec/IPSec) headers
- Updating IP header checksum
- Updating the TCP checksum
- Updating Ethernet CRC
- Updating the authentication fields

Figure 7 shows the packet editing/modify operations for MACSec and IPSec. Many modifications might be needed in the packet header fields, including calculations and insertion of checksum and CRC, before sending the packet out at the egress port. Along with standard headers, packets can often include proprietary packet headers and can also require packet encapsulation and decapsulation with different protocol headers (VXLAN, IP in IP, GRE, etc.). Xilinx's programmable devices can most flexibly implement packet modification at the line rate.

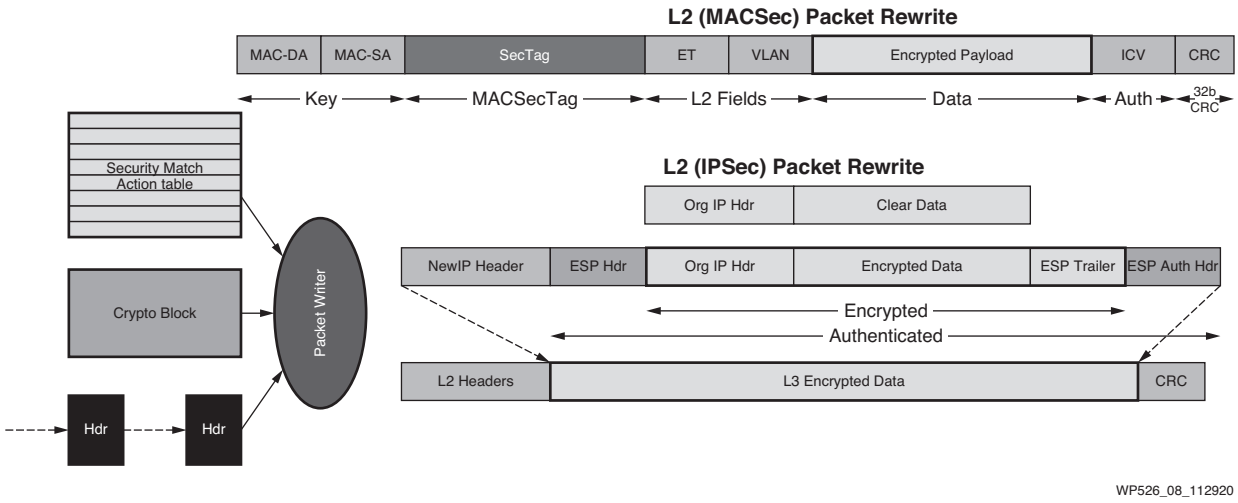


*Figure 7:* **Packet Rewrite for MACSec and IPSec Packets**

Xilinx devices are also P4 programmable, so packet rewrite operations can also be implemented using P4. P4 deparser functionality can further simplify building and inserting the header compared to RTL implementation. The P4 editor code can be synthesized using the Xilinx P4 compiler to work at the line rate on Xilinx devices.

For application layer security implementation, the requirements for packet rewrite operations are more complex. For example, if TCP packets are terminated inside the FPGA, then session tracking and encapsulation/decapsulation requirements are more logic and memory intensive than IPSec or MACSec packet modification requirements.

Packet modification tasks can also be performed in the software running on the CPU cores, but the throughputs needed for high-end security appliances cannot meet the line rate operation with software implementation. Another important advantage of performing the packet processing operations in the programmable hardware is that it can save a lot of CPU resources (CPU cores), which can otherwise be allocated for real applications running in software.

# Application-Level Security Processing in FPGAs

FPGAs are ideal for inline security processing in next-generation firewalls because the requirements of higher performance, flexibility, and low latency operations are successfully met using FPGAs. FPGAs can also implement application-level security functions to further save compute resources and increase performance.

Some common examples of application security processing in FPGAs include:

- TCP offload engine
- Regular expression matching
- Asymmetric Crypto (PKI) processing
- TLS processing

Since many user space applications use TCP as the communication protocol in client or server mode and TCP is the basic block for secure (TLS/SSL) connections between client and server, the TCP offload engine (TOE) is an important block to be offloaded for inline FPGA processing. Enterprise firewalls usually terminate millions of TCP connections simultaneously, which consumes lots of CPU cycles and memory. To implement application-level security processing, expensive high-end CPUs with a large number of cores might be required to terminate the millions of TCP/UDP connections. TCP processing implementation in FPGAs can save significant cost and power by saving the number of cores required to implement TOE.

Figure 8 shows an example of FPGA-assisted packet processing in security appliances. Since the packet entering at the firewall's network interface might belong to many different applications, tracking the packets associated with multiple applications and sending or receiving them to the right application is a memory-intensive stateful operation. This association also requires re-ordering, segmentation, and reassembly of the TCP segments. While the new connection request and authentication of the protocol messages can still be handled by the CPU, the FPGAs can track the active sessions and assign the packets to relevant applications based on the session ID.

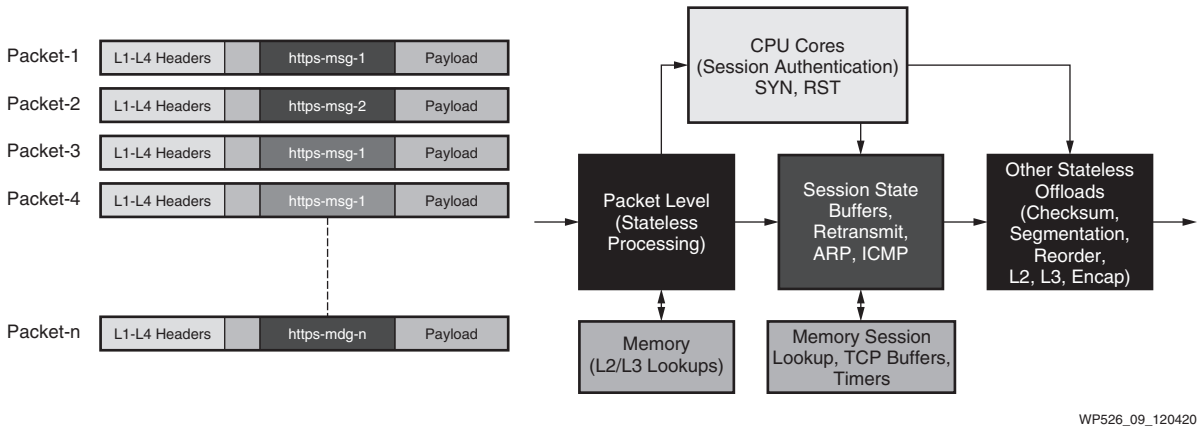


*Figure 8:* **Stateful Application-Level Processing Using FPGAs**

## TLS Offloads/Processing in FPGAs

The TLS processing in FPGAs is an extension of the TCP offload engine, where encryption and decryption of the TCP payload is performed in the FPGA. The initiation and authentication of the TLS session happens in the software (CPU). When a secure connection is established, the subsequent TLS record processing is handled by the FPGA.

Figure 9 shows the components of a complete inline SSL processing as a CPU offload in a Xilinx device. The Xilinx device implements the entire logic to process the Ethernet packets arriving at the 100G interface. It identifies the TCP and TLS flows and accordingly directs the packets to the CPU or processes them using programmable resources.

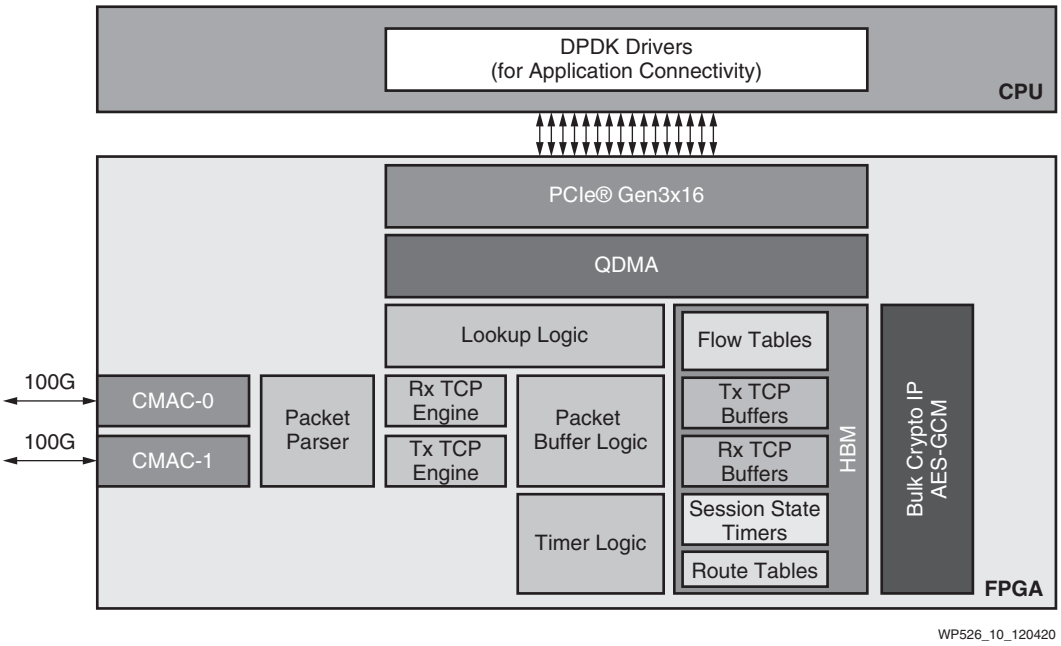


*Figure 9:* **TLS Offload in FPGA**

## Regular Expression (Regex) in FPGA

Regular expression (regex) involves matching strings or special characters in the payload data of a traffic flow. It is widely used with DPI, IPS/IDS, DLP, and DDoS mitigation. Regex matching is commonly done in software, using a dedicated software library. Since a regex search involves matching the payload against millions of rules, software-only regex processing poses performance and latency challenges for next-generation security appliances.

Figure 10 shows 100Gb/s inline regex processing using a Xilinx device. In this regex acceleration processing model, Perl compatible regular expression (PCRE) or snort rules are first compiled in the software compiler and then sent to the FPGA connected to the CPU via PCI interface as binary string matching rule entries to be stored in the FPGA's internal SRAM or DRAM (HBM or DDR) memory. FPGAs can populate millions of regex rules/entries (combination of special characters and words converted to binary) in internal SRAM or DRAM (on-chip HBM or external DDR). Inline acceleration of regex processing results in a significant performance gain (10X–30X) against the software.

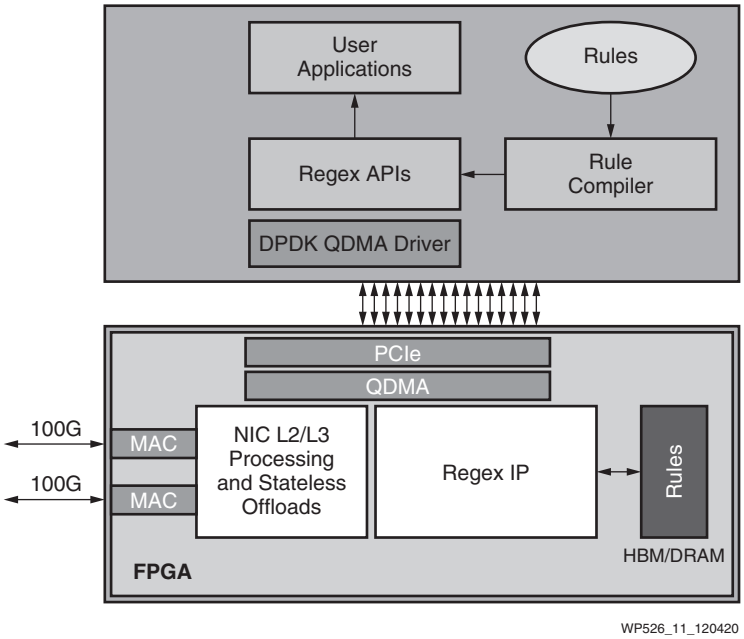


*Figure 10:* **Regex Matching in FPGAs**

## Machine Learning (ML) in FPGA-based Security Appliances

In next-generation security appliances, ML-based traffic analysis and malware detection is one of the key applications. ML models will be deployed to inspect encrypted traffic by analyzing the specific patterns in the encrypted data. In high-end security appliances, the ML model is required to process huge amounts of real-time data to predict anomalies, so use of the accelerators to implement ML models will greatly benefit high throughput and low latency malware prediction. Firewalls have already started deploying ML models in software for anomaly detection. In next-generation appliances, Xilinx's programmable devices will offer much faster prediction rates by offloading the ML models to the programmable logic.

These FPGA-based ML models can include:

- Random trees (Random Forest)
- Deep neural networks (DNN)
    - Multi-layer perceptron (MLP) or convolution neural networks (CNN)

The selection of the inference model depends on various factors such as accuracy, frequency of change in input patterns, training requirements, FPGA resource utilization, etc.

Xilinx ML solutions include software libraries and tools that work with most common ML frameworks. These models are efficiently mapped into lookup tables (LUTs), DSPs, SRAM/DRAM memory on Xilinx's programmable devices—and in addition, in the AI Engines available in Versal ACAPs.

Another advantage of implementing ML model for security analytics in the FPGA is that inline traffic/packet processing needed for malware prediction is on the same FPGA. In the inline

implementation of the ML model, having network interfaces connected on the same FPGA saves the PCIe® bandwidth needed for sending data from the CPU to the ML models.

Figure 11 shows the use of an ML model for 200Gb/s firewall. The TLS processor has the TLS parser and extraction of the TLS parameters in IP datagrams. These parameters are then fed to the ML processor to look up and adjust the coefficients of the ML model. Based on the coefficient, the model predicts a good vs. the bad signature in TLS traffic.

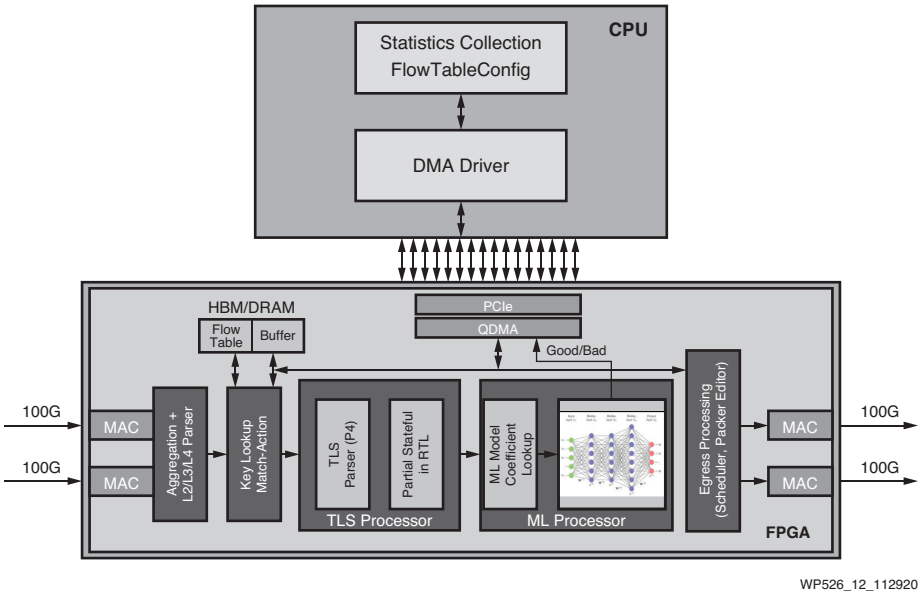


*Figure 11:* **ML Model in Security Appliances**

# Next-Generation Security Technologies Using FPGAs

## Post Quantum Cryptography

Many of the currently known asymmetric algorithms are susceptible to being broken by quantum computers. The study and implementation of quantum computing safe cryptographic algorithms has already started, and the use of FPGAs for implementing such algorithms is already described in academic papers. Asymmetric security algorithms such as RSA-2K, RSA-4K, ECC-256, DH, and ECCDH are most impacted by quantum computing. The new asymmetric algorithm implementation and NIST standardization discussions are currently in progress.

The currently proposed post quantum cryptography (PQC) includes Ring-learning with error (R-LWE) algorithms for:

- Public key encryption (PKC)
- Digital signature
- Key establishment

Implementation of the proposed public key encryption includes some well-known mathematical operations (TRNG, Gaussian Noise sampler, Polynomial addition, Binary Polynomial scaler division,

multiplication, etc.). FPGA IPs for many of these algorithms already exist or can be efficiently implemented using FPGA building blocks such as DSPs and AI Engines in current and next-generation Xilinx devices.

## Secure Access Service Edge (SASE)

Secure Access Service Edge (SASE) is an emerging next-generation enterprise security technology to address the need of dynamic secure access for organizations. The early definition of SASE integrates the adaptable network and security requirements at the enterprise edge, which includes SD-WAN, software and physical firewalls, and web security gateways. SASE requires a dynamic security policy update to provide secure and uninterrupted access to connecting applications.

Implementation of SASE in hardware using FPGAs is still in the early phase, but since FPGAs are fully programmable, they can still play a role in flow processing and providing dynamic secure connectivity pipes through L2/L3/L4 encryption techniques and other technologies described above.

## Xilinx Tools and IPs for Security Appliances

Xilinx devices have high-performance programmable resources with state-of-the-art tools and IPs that are ideal for designing and implementing security processing for network traffic. They provide the highest data and signal processing throughput with the latest multirate high throughput SerDes for designing with the latest interface standards, which include 1G-400G Ethernet, 600G Interlaken, and up to 400G PCIe throughputs. Xilinx devices also provide registered inter-die routing lines enabling up to 600MHz programmable logic operation.

Along with basic high-performance design resources, Xilinx also provides multiple design IPs for security processing. These programmable IPs include MAC interface, high-speed DMA for data transfer to/from host, search IPs (BCAM, TCAM, and STCAM) for traffic classification and routing, and on-chip HBM and/or DDR memory interfaces and soft crypto engine (SCE) for bulk encryption using AES-GCM cipher.

Xilinx also has an ecosystem of partners who provide end-to-end solutions for bulk crypto, using multiple cipher protocols, and IPs for asymmetric crypto using most common key exchanges (ECCDH, RSA-2K, RSA-4K, etc.) In addition to the base level standard crypto IPs from partners, Xilinx is currently working with partners for implementation of advanced level (L4+) security IP, which includes:

- TCP offload engines with millions of active sessions
- Inline SSL offload reference design
- Application-level security offloads (5G L2 acceleration)
- Packet processing for 10K+ IPSec sessions

Xilinx's latest generation devices (Versal™ Premium ACAPs) have a hardened High-Speed Crypto Engine (HSC) that can be used as a crypto engine for implementing MACSec, IPSec, or SSL processing of up to 400Gb/s using the AES-GCM protocol. Each HSC Engine can support MACsec, IPSec, and any other bulk crypto requirements with 1x400G, 2x200G, or 4x100G channelized modes

with up to 128 security associations (SA) per 100G. Additional SAs can be implemented with programmable logic.

# Summary

As communication networks (edge, access, and core) are transforming towards higher performance with application-level policy awareness, the requirement for security processing at higher throughput has significantly increased. Also with the upgrade of access technologies and deployment of 5G access (xHaul), next-generation PON and cable networks, the number of devices connected at the network aggregation will rise exponentially. In a next-generation network security appliance, 2X-4X throughput for L2 (MACSec) security and L3 (IPSec) security processing are required. Also, next-generation networks are more intent and policy-based, so demand for high throughput application-level security processing (L4-L7 security) has gone up significantly.

High throughput application security implementation requires high throughput packet processing and significant compute resources for ciphering requirements. The software-only application security implementation cannot meet the performance and latency expectations. The latency requirements are even more critical for 5G low latency applications, therefore, the use of an programmable accelerators as an inline security processor is becoming increasingly important in next-generation security appliances.

In addition to the throughput and latency problems addressed by using Xilinx devices in next-generation firewalls, other advantages include the implementation of new technologies such as machine learning (ML) models, secure access service edge (SASE), and post quantum crypto (PQC). Xilinx devices provide an ideal platform of choice for hardware acceleration for these technologies because software-only implementation is unable to meet the performance requirements. Xilinx is continuously developing and upgrading the IPs, tools, software, and reference designs for current and next-generation network security solutions.

## Acknowledgment

*The following Xilinx employee has authored or contributed to this white paper:*
*Awanish Verma, principle architect and director of technical marketing.*

# Revision History

The following table shows the revision history for this document:

| Date | Version | Description of Revisions |
|---|---|---|
| 03/22/2021 | 1.0 | Initial Xilinx release. |

# Disclaimer



# Automotive Applications Disclaimer